Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»



**Отчет**

**Лабораторная работа № 3**

**По курсу «Разработка интернет-приложений»**

**«Функциональные возможности языка Python»**

**ИСПОЛНИТЕЛЬ:**

Группа ИУ5-55

\_Кожуро Б.Е.

"20"\_октября\_2020 г.

**ПРЕПОДАВАТЕЛЬ:**

\_\_\_Гапанюк Ю.Е.\_\_

"\_\_"\_\_\_\_\_\_\_\_2020 г.

Москва 2020

1. **Задание**
   1. Необходимо для произвольной предметной области реализовать три шаблона проектирования: один порождающий, один структурный и один поведенческий. В качестве справочника шаблонов можно использовать следующий каталог.
   2. Для каждой реализации шаблона необходимо написать модульный тест. В модульных тестах необходимо применить следующие технологии:

* TDD – фреймворк
* BDD – фреймворк
* Создание Mock-объектов

1. **Текст программы**
   1. **main.py**

from abc import ABC, abstractmethod  
  
  
# порождающий паттерн проектирования  
# абстрактная фабрика  
# предметная область:  
  
class Motherboard(ABC):  
 @abstractmethod  
 def install (self):  
 pass  
  
class Processor(ABC):  
 @abstractmethod  
 def install (self, motherboard\_type):  
 pass  
  
class GPU(ABC):  
 @abstractmethod  
 def install (self):  
 pass  
  
class PC\_Factory(ABC):  
 @abstractmethod  
 def install\_motherboard(self):  
 pass  
  
 @abstractmethod  
 def install\_processor(self):  
 pass  
  
 @abstractmethod  
 def install\_gpu(self):  
 pass  
  
 @abstractmethod  
 def price(self):  
 pass  
  
class Intel\_Motherboard(Motherboard):  
 def install (self):  
 return "установлена плата с чипсетом INTEL"  
 m\_type = "I"  
 price = 110  
  
class AMD\_Motherboard(Motherboard):  
 def install(self):  
 return "установлена плата с чипсетом AMD"  
 m\_type = "A"  
 price =100  
  
class Intel\_Processor(Processor):  
 def install(self, motherboard\_type):  
 if motherboard\_type.m\_type == "I":  
 return "установлен процессор INTEL"  
 else:  
 return "неверный тип платы"  
 price = 200  
  
class AMD\_Processor(Processor):  
 def install(self, motherboard\_type):  
 if motherboard\_type.m\_type == "A":  
 return "установлен процессор AMD"  
 else:  
 return "неверный тип платы"  
 price =180  
  
  
class Nvidia\_GPU(GPU):  
 def install (self):  
 return "установлена видеокарта Nvidia"  
 price =220  
  
class AMD\_GPU (GPU):  
 def install(self):  
 return "установлена видеокарта AMD"  
 price =200  
  
class Intel\_Nvidia(PC\_Factory):  
 def install\_motherboard(self):  
 return Intel\_Motherboard()  
 def install\_processor(self):  
 return Intel\_Processor()  
 def install\_gpu(self):  
 return Nvidia\_GPU()  
 def price (self):  
 return Intel\_Motherboard.price+Intel\_Processor.price+Nvidia\_GPU.price  
  
class Intel\_AMD(PC\_Factory):  
 def install\_motherboard(self):  
 return Intel\_Motherboard()  
 def install\_processor(self):  
 return Intel\_Processor()  
 def install\_gpu(self):  
 return AMD\_GPU()  
 def price (self):  
 return Intel\_Motherboard.price+Intel\_Processor.price+AMD\_GPU.price  
  
  
class AMD\_AMD(PC\_Factory):  
 def install\_motherboard(self):  
 return AMD\_Motherboard()  
 def install\_processor(self):  
 return AMD\_Processor()  
 def install\_gpu(self):  
 return AMD\_GPU()  
 def price (self):  
 return AMD\_Motherboard.price+AMD\_Processor.price+AMD\_GPU.price  
  
  
def client\_code(factory):  
 mb = factory.install\_motherboard()  
 proc = factory.install\_processor()  
 gpu = factory.install\_gpu()  
  
 print (mb.install())  
 print (proc.install(mb))  
 print (gpu.install())  
 print (factory.price())  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
  
 client\_code(AMD\_AMD())  
 print('\n')  
 client\_code(Intel\_Nvidia())  
 print('\n')  
 client\_code(Intel\_AMD())

* 1. **Tests\_AF.py**

import unittest  
from unittest import TestCase  
from unittest.mock import patch  
from main import Intel\_AMD  
from main import AMD\_AMD  
from main import Intel\_Nvidia  
from main import PC\_Factory  
from main import AMD\_Motherboard  
  
class PC\_FactoryTestCase(TestCase):  
  
 #тест логики определения неправильного типа платы  
 @patch('main.Intel\_Nvidia.install\_motherboard', return\_value=AMD\_Motherboard())  
 def test\_intel\_nvidia\_build(self, arg):  
 factory = Intel\_Nvidia()  
 proc = factory.install\_processor()  
 mb = factory.install\_motherboard()  
 self.assertEqual("неверный тип платы", proc.install(mb))  
  
 #проверка фабрики Intel\_AMD установка процессора  
 def test\_intel\_AMD\_build\_proc(self):  
 factory = Intel\_AMD()  
 proc = factory.install\_processor()  
 mb = factory.install\_motherboard()  
 self.assertEqual("установлен процессор INTEL", proc.install(mb))  
  
 # проверка фабрики Intel\_AMD установка платы  
 def test\_intel\_AMD\_build\_MB(self):  
 factory = Intel\_AMD()  
 mb = factory.install\_motherboard()  
 self.assertEqual("установлена плата с чипсетом INTEL", mb.install())  
  
 # проверка фабрики Intel\_AMD установка видеокарты  
 def test\_intel\_AMD\_build\_GPU(self):  
 factory = Intel\_AMD()  
 gpu = factory.install\_gpu()  
 self.assertEqual("установлена видеокарта AMD", gpu.install())  
  
 # проверка фабрики Intel\_AMD подсчет цены  
 def test\_intel\_AMD\_build\_Price(self):  
 factory = Intel\_AMD()  
 mb =factory.install\_motherboard()  
 pr = factory.install\_processor()  
 gpu = factory.install\_gpu()  
 mb.install()  
 pr.install(mb)  
 gpu.install()  
 self.assertEqual(510, factory.price())  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

* 1. **adapter.py**

# Итератор для удаления дубликатов  
class Unique(object):  
 def \_\_init\_\_(self, items, \*\*kwargs):  
 self.position = 0  
 self.unique\_elements = set()  
 self.elements = items  
 if len(kwargs) != 0:  
 self.ignore\_case = kwargs  
 else:  
 self.ignore\_case = False  
  
 def \_\_next\_\_(self):  
 while True:  
 for element in self.elements:  
 current\_element = element  
 self.position = self.position + 1  
 if isinstance(current\_element, str):  
 if not (current\_element in self.unique\_elements) and \  
 not (self.ignore\_case and (current\_element.lower() in self.unique\_elements)):  
 if self.ignore\_case:  
 self.unique\_elements.add(current\_element.lower())  
 else:  
 self.unique\_elements.add(current\_element)  
 return current\_element  
 elif (current\_element not in self.unique\_elements):  
 self.unique\_elements.add(current\_element)  
 return current\_element  
 else:  
 raise StopIteration  
  
 def \_\_iter\_\_(self):  
 return self  
  
# Тест  
def main():  
 array = ['a', 'a', 'a' , 'A' ,'A', 'b', 'B', 10, 1, 1, 'Boris', 'boris', 'bOrIs', 'Boria']  
 iterator1 = Unique(array)  
# структурный паттерн проектирования  
# адаптер  
# предметная область: проверка, входит ли кпрямоугольная деталь в круглое отверстие.  
  
  
# класс круглых деталей  
from math import sqrt  
  
  
class circle:  
 def \_\_init\_\_(self, r):  
 self.r = r  
 def get\_r(self):  
 return self.r  
  
  
# класс прямокгольниак  
class area:  
 def \_\_init\_\_(self, x, y):  
 self.x = x  
 self.y = y  
  
 def get\_x(self):  
 return self.x  
 def get\_y(self):  
 return self.y  
  
  
# класс отверстия  
class roundhole:  
  
 def \_\_init\_\_(self, r):  
 self.r = r  
  
 def get\_r(self):  
 return self.r  
  
 def fits(self, detail):  
  
 if self.get\_r() >= detail.get\_r():  
 return f"влезает , радиус детали= {detail.get\_r()} и отверстия={self.get\_r()}"  
 else:  
 return f"не влезает , радиус детали= {detail.get\_r()} и отверстия={self.get\_r()}"  
  
  
# адаптер  
class adapter(area):  
  
 def \_\_init\_\_(self, area):  
 self.area = area  
  
 def get\_r(self):  
 return sqrt((self.area.get\_x() / 2)\*\*2 + (self.area.get\_y() / 2)\*\*2)  
  
  
def client\_code():  
 circle1 = circle(10)  
 circle2 = circle(20)  
 area1 = area(8, 9)  
 area2 = area(20, 16)  
 hole = roundhole(12)  
 print(hole.fits(circle1))  
 print(hole.fits(circle2))  
 print(hole.fits(adapter(area1)))  
 print(hole.fits(adapter(area2)))  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
  
 client\_code()

* 1. **Wrapper.py**

from behave import \*  
from adapter import circle  
from adapter import roundhole  
from adapter import area  
from adapter import adapter  
  
  
@given('circle r={circle\_r}, hole r={hole\_r}')  
def step(context, circle\_r, hole\_r):  
 context.circle = circle(int(circle\_r))  
 context.hole = roundhole(int(hole\_r))  
  
  
@given('area x={x}, y={y}, hole r={hole\_r}')  
def step(context, x, y, hole\_r):  
 context.area = area(int(x), int(y))  
 context.hole = roundhole(int(hole\_r))  
  
  
@then('Проходит')  
def step(context):  
 assert context.hole.fits(context.circle) == f"влезает , радиус детали= {context.circle.get\_r()} " \  
 f"и отверстия={context.hole.get\_r()}"  
  
  
@then('Не проходит')  
def step(context):  
 assert context.hole.fits(context.circle) == f"не влезает , радиус детали= {context.circle.get\_r()} " \  
 f"и отверстия={context.hole.get\_r()}"  
  
  
@then('С адаптером прямоугольник проходит')  
def step(context):  
 context.adapted = adapter(context.area)  
 assert context.hole.fits(context.area) == f"влезает , радиус детали= {context.adapted.get\_r()} " \  
 f"и отверстия={context.hole.get\_r()}"  
  
  
@then('С квадратом прямоугольник не проходит')  
def step(context):  
 context.adapted = adapter(context.area)  
 assert context.hole.fits(context.adapter) == f"не влезает , радиус детали= {context.adapted.get\_r()} " \  
 f"и отверстия={context.hole.get\_r()}"

* 1. **Tutorial.feature.py**

Feature: Без адаптера  
 Scenario: Проходит  
 Given circle r=**10**, hole r=**12** Scenario: Не проходит  
 Given circle r=**15**, hole r=**12** Scenario: Проходит с адаптером  
 Given area x=**4**, y=**5**, hole r=**12** Scenario: Не проходит с адаптером  
 Given area x=**20**, y=**18**, hole r=**12**

* 1. **Observer.py**

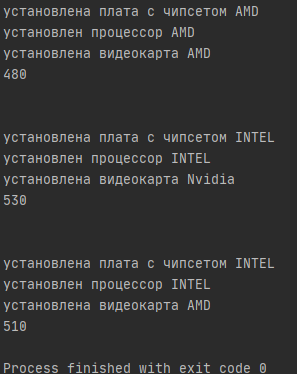
# поведенческий паттерн проектирования  
# наблюдатель  
# предметная область: оповещение об изменнении температуры на метеостанции.  
  
import numpy  
  
  
class subscriber():  
 def \_\_init\_\_(self, name):  
 self.name = name  
  
  
class meteostation():  
 def \_\_init\_\_(self, name):  
 self.name = name  
 self.temperature = [0]  
 self.subscribers = list()  
  
 def subscribe(self, sub):  
 self.subscribers.append(sub)  
 return f"{sub.name} подписался"  
  
 def notify(self):  
 for sub in self.subscribers:  
 print(  
 f"Оповещаю {sub.name} об изменении температуры на {self.name} с {self.temperature[-2]} до {self.temperature[-1]}")  
 return "оповещены подписчики"  
  
 def measure(self):  
 self.temperature.append(numpy.random.randint(-2, 2))  
 if self.temperature[-2] != self.temperature[-1]:  
 self.notify()  
 return "оповещение запущено"  
 return "оповещение не нужно"  
  
  
def client\_code():  
 station1 = meteostation("станция 1")  
 station2 = meteostation("станция 2")  
 subscriber1 = subscriber("Ann")  
 subscriber2 = subscriber("John")  
 station1.subscribe(subscriber1)  
 station2.subscribe(subscriber1)  
 station2.subscribe(subscriber2)  
 station1.measure()  
 station1.measure()  
 station2.measure()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 client\_code()

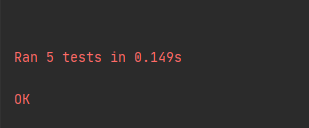
* 1. **Tests\_observer.py**

import unittest  
from unittest import TestCase  
from unittest.mock import patch  
from observer import meteostation  
from observer import subscriber  
  
  
class ObserverTestCase(TestCase):  
  
 #тест подписки  
 def testsubscribe(self):  
 st1 = meteostation("st1")  
 sub1 = subscriber("S1")  
 self.assertEqual("S1 подписался", st1.subscribe(sub1))  
 #тест запуска оповещения  
 def testnotify(self):  
 st1 = meteostation("st1")  
 st1.temperature = [-4]  
 sub1 = subscriber("s1")  
 self.assertEqual(st1.measure(), "оповещение запущено")  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

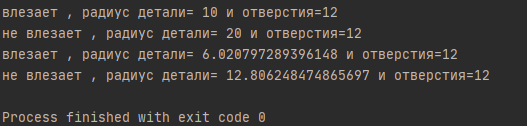
1. **Скриншоты выполнения**

**Абстрактная фабрика:**

****

****

**Адаптер:**

****

##teamcity[testCount count='4']

##teamcity[enteredTheMatrix]

##teamcity[testSuiteStarted name='Без адаптера' locationHint='file:///C:\Users\ksarb\PycharmProjects\pythonProject\feartures\feartures\tutorial.feature:1']

##teamcity[testSuiteStarted name='Проходит' locationHint='file:///C:\Users\ksarb\PycharmProjects\pythonProject\feartures\feartures\tutorial.feature:2']

##teamcity[testStarted name='Given circle r=10, hole r=12' locationHint='file:///C:\Users\ksarb\PycharmProjects\pythonProject\feartures\feartures\tutorial.feature:3']

##teamcity[testFinished timestamp='2020-11-15T17:12:53.402' duration='0' name='Given circle r=10, hole r=12']

##teamcity[testSuiteFinished timestamp='2020-11-15T17:12:53.402' name='Проходит']

##teamcity[testSuiteStarted name='Не проходит' locationHint='file:///C:\Users\ksarb\PycharmProjects\pythonProject\feartures\feartures\tutorial.feature:4']

##teamcity[testStarted name='Given circle r=15, hole r=12' locationHint='file:///C:\Users\ksarb\PycharmProjects\pythonProject\feartures\feartures\tutorial.feature:5']

##teamcity[testFinished timestamp='2020-11-15T17:12:53.404' duration='0' name='Given circle r=15, hole r=12']

##teamcity[testSuiteFinished timestamp='2020-11-15T17:12:53.404' name='Не проходит']

##teamcity[testSuiteStarted name='Проходит с адаптером' locationHint='file:///C:\Users\ksarb\PycharmProjects\pythonProject\feartures\feartures\tutorial.feature:6']

##teamcity[testStarted name='Given area x=4, y=5, hole r=12' locationHint='file:///C:\Users\ksarb\PycharmProjects\pythonProject\feartures\feartures\tutorial.feature:7']

##teamcity[testFinished timestamp='2020-11-15T17:12:53.406' duration='1' name='Given area x=4, y=5, hole r=12']

##teamcity[testSuiteFinished timestamp='2020-11-15T17:12:53.406' name='Проходит с адаптером']

##teamcity[testSuiteStarted name='Не проходит с адаптером' locationHint='file:///C:\Users\ksarb\PycharmProjects\pythonProject\feartures\feartures\tutorial.feature:8']

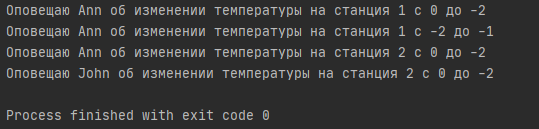
##teamcity[testStarted name='Given area x=20, y=18, hole r=12' locationHint='file:///C:\Users\ksarb\PycharmProjects\pythonProject\feartures\feartures\tutorial.feature:9']

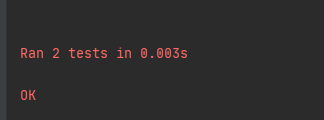
##teamcity[testFinished timestamp='2020-11-15T17:12:53.407' duration='0' name='Given area x=20, y=18, hole r=12']

##teamcity[testSuiteFinished timestamp='2020-11-15T17:12:53.407' name='Не проходит с адаптером']

##teamcity[testSuiteFinished timestamp='2020-11-15T17:12:53.407' name='Без адаптера']

**Наблюдатель:**

****

****